EECS 203: Discrete Mathematics
Winter 2024
Homework 11

Due **Tuesday, April 23**, 10:00 pm

No late homework accepted past midnight.

Number of Problems: $6 + 0$ Total Points: $100 + 0$

- **Match your pages!** Your submission time is when you upload the file, so the time you take to match pages doesn't count against you.
- Submit this assignment (and any regrade requests later) on Gradescope.
- Justify your answers and show your work (unless a question says otherwise).
- By submitting this homework, you agree that you are in compliance with the Engineering Honor Code and the Course Policies for 203, and that you are submitting your own work.
- Check the syllabus for full details.

Individual Portion

1. Big-*O*reo [15 points]

Give the tightest big-O estimate for each of the following functions. Justify your answers.

(a) $f(n) = (2^n + n^n) \cdot (n^3 + n \log n^n)$

(b) $g(n) = (n^n + n!) \cdot (n + 1) \quad + \quad (n^3 + 3^n) \cdot (\sqrt{n} + \log n)$

(c) $h(n) = (n^n + n^2) \cdot (n^n + n) \quad + \quad (\log 3 + n^n) \cdot (n^2 + n^n)$

Solution:

(a) n^n and n^3 dominate respectively. Thus

$$f(n) = O(n^n n^3) = O(n^{n+3})$$

(b) In the left product, n^n and n dominate. In the right product, 3^n and $\sqrt{n}$ dominate. But the left product will dominate over the right product. So

$$g(n) = O(n^n n) = O(n^{n+1})$$

(c) In both products, n^n and n^n dominate. So $2n^{2n}$ will dominate the sum. So

$$h(n) = O(n^{2n})$$

.

2. On the Run [20 points]

Give the tightest big-O estimate for the number of operations (where an operation is arithmetic, a comparison, or an assignment) used in each of the following algorithms. **Explain your reasoning.**

(a)
```
function doubleTrouble(a_1, ..., a_N ∈ ℝ, j ∈ ℝ)
    j ← 1
    for i := 1 to N do
        if i = j then
            j ← 2j
        end if
    end for
    return j
```

```
    end function
```

(b)
```
    function sumSquares(N ∈ ℤ⁺)
        if N = 1 then
            return 1
        end if
        value ← sumSquares(N − 1) + N²
        return value
    end function
```

(c)
```
    function findLTMinProduct(a_1, ..., a_N ∈ ℝ)
        p ← 203
        for i ≔ 1 to N do
            for j ≔ 1 to N do
                if a_i a_j < p then
                    p ← a_i a_j
                end if
            end for
        end for
        numLTMinProduct ← 0
        for k ≔ 1 to N do
            if a_k < p then
                numLTMinProduct ← numLTMinProduct + 1
            end if
        end for
        return numLTMinProduct
    end function
```

(d)
```
    function subtractAndAdd(N ∈ ℤ)
        while N > 0 do
            if N is even then
                N ← N − 3
            end if
            if N is odd then
                N ← N + 1
            end if
        end while
        return N
    end function
```

(e)
```
    function search(a_1, ..., a_N ∈ ℝ, target ∈ ℝ)
        left ← 1
        right ← N
        while True do
            mid ← ⌊(left+right)/2⌋
```

```
        if a_mid = target then
            return mid
        end if
        if right ≤ left then
            return −1
        end if
        if a_mid < target then
            left ← mid + 1
        end if
        if a_mid > target then
            right ← mid − 1
        end if
    end while
end function
```

Solution:

(a) The loop executes N times, and contains no loops within itself. So this function is $O(N)$.

(b) The function calls itself recursively with $N - 1$ until it reaches 1. So it is $O(N)$.

(c) The first loop executes N times, containing another loop which executes N times, for a total of N^2. The last loop executes N times. So the N^2 dominates, and the function is $O(N^2)$.

(d) This loop will take at most $\frac{N+1}{2}$ steps, if N is odd. (The first loop will add one. Then, every subsequent loop will do -3, +1 = -2.) If N is even, it will take at most $\frac{N}{2}$ steps. (Every loop will do -3, +1 = -2.) So this function will be $O(N)$.

(e) The loop ends when $a_{mid} = target$ or $right \leq left$. Every loop, the midpoint is set to the floor of the midpoint between the left and right indexes. If the target is not found, the left or right index moves next to the midpoint, and the loop runs again. So every loop, the search area is halved. Thus the loop will take at most $\log_2 N + C$ steps. So the time complexity will be $O(\log N)$.

3. This one's bound to be fun! [18 points]

You are given the following bounds on functions f and g:

- $f(x)$ is $O(203^x x^2)$ and $\Omega(3^x \log x)$

- $g(x)$ is $O(\frac{x!}{2^x})$ and $\Omega(4^x)$

Find the following, simplify your answer as much as possible.

(a) Find the tightest big-O and big-Ω estimates that can be *guaranteed* of $f(x)(g(x))^2$.

(b) Find the tightest big-O and big-Ω estimates that can be *guaranteed* of $f(x) + g(x)$.

(c) Let $h(x) = f(x) - g(x)$. Prove or disprove that $h(x)$ is $\Omega(4^x)$.

Solution:

(a) For big-O, we find the parts which will dominate the function. This will be

$$f(x)(g(x))^2 = O(203^x x^2) \cdot O\left(\frac{x!}{2^x}\right)^2 = O\left(\left(\frac{203}{4}\right)^x (x!)^2 x^2\right)$$

For Ω, we compute using the Ω portions of the respective functions.

$$f(x)(g(x))^2 = \Omega(3^x \log x) \cdot \Omega(4^x) = \Omega(12^x \log x)$$

(b) Since this is addition, we simply select the dominating parts between each added function. This will be $O\left(\frac{x!}{2^x}\right)$ and $\Omega(4^x)$.

(c) By the rule of scalar multiplication, the scalar multiplication by -1 does not change the Ω of $g(x)$. Then, by the addition rule, we take the dominating Ω of f and g, which is 4^x. So by big-Ω properties, $h(x) = \Omega(4^x)$.

4. Big Function Fun [16 points]

Prove or disprove the following:

(a) If $f(x)$ is $O(g(x))$ then $2^{f(x)}$ is $O(2^{g(x)})$.

(b) If $f(x)$ is $O(g(x))$ then $(f(x))^2$ is $O\big((g(x))^2\big)$.

Note that in these proofs you do not need to use the definition of big-O, but can use the properties for combining mathematical functions covered in lecture.

Solution:

(a) This is true. Assume $f(x) = O(g(x))$. By the definition of big-O, there exists $c_1, k \in \mathbb{R}$ such that for all $x \geq k$, $f(x) \leq c_1 g(x)$. If $f(x) \leq c_1 g(x)$, then

$$2^{f(x)} \leq 2^{c_1 g(x)} = 2^{c_1} 2^{g(x)}$$

So $2^{f(x)} = O(2^{g(x)})$ with $c = 2^{c_1}$ and $k = k$.

(b) By the product property for big-O, $(f_1 \cdot f_2)(x) = O(g_1(x) \cdot g_2(x))$ for $f_1(x) = O(g_1(x))$ and $f_2(x) = O(g_2(x))$. So the statement is true if we take $f_1(x) = f_2(x) = f(x)$, $g_1(x) = g_2(x) = g(x)$ in the product property.

5. Roots and Shoots [16 points]

Suppose f satisfies $f(n) = 2f(\sqrt{n}) + \log_2 n$, whenever n is a perfect square greater than 1, and additionally satisfies $f(2) = 1$.

(a) Find $f(16)$.

(b) Find a big-O estimate for $g(m)$ where $g(m) = f(2^m)$.

Hint: Make the substitution $m = \log_2 n$.

(c) Find a big-O estimate for $f(n)$.

Solution:

(a)

$$\begin{aligned} f(16) &= 2f(4) + \log_2 16 \\ &= 4f(2) + 2\log_2 4 + 4 \\ &= 4 + 4 + 4 \\ &= 12 \end{aligned}$$

(b) Substituting $\log_2 n = m$,

$$\begin{aligned} g(m) &= f(2^m) \\ &= 2f(2^{m/2}) + m \\ &= 2g\left(\frac{m}{2}\right) + m \end{aligned}$$

By the Master Theorem, this recursive relation is $O(m \log m)$.

(c) Using the previous part and substituting $m = \log_2 n$, we find

$$f(n) = O\left(\log_2(n) \log(\log_2 n)\right)$$

6. GG Brown Laboratory [15 points]

What is the tightest big-O bound on the runtime complexity of the following algorithm?

```
function BADSEARCH(n)
    if n ≥ 1 then
        BADSEARCH(⌊n/3⌋)
        for i := 1 to n do
            for j := 1 to ⌊n/2⌋ do
                print "Hello I am lost"
            end for
        end for
        BADSEARCH(⌊n/3⌋)
        print "Nevermind I got it"
    end if
end function
```

Solution:

This function does not run any commands if n is greater than 1. If it is greater than 1, then it will run itself with $n/3$ twice, and also run the middle loops, which run a total of $n^2/2$ commands at most. So we have the recursive relation $B(n)$ for the number of commands which the function executes:

$$B(n) = 2B\left(\frac{n}{3}\right) + \frac{n^2}{2}$$

Then by the Master Theorem, this function is $\boxed{O(n^2)}$, since $2 < 3^2$.